Peer-to-Peer Networks
13 Internet – The Underlay Network

Christian Ortolf
Technical Faculty
Computer-Networks and Telematics
University of Freiburg
# Types of Networks

<table>
<thead>
<tr>
<th>Interprocessor distance</th>
<th>Processes located in same</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 m</td>
<td>Square meter</td>
<td>Personal area network</td>
</tr>
<tr>
<td>10 m</td>
<td>Room</td>
<td>Local area network</td>
</tr>
<tr>
<td>100 m</td>
<td>Building</td>
<td>Metropolitan area network</td>
</tr>
<tr>
<td>1 km</td>
<td>Campus</td>
<td>Wide area network</td>
</tr>
<tr>
<td>10 km</td>
<td>City</td>
<td>The Internet</td>
</tr>
<tr>
<td>100 km</td>
<td>Country</td>
<td></td>
</tr>
<tr>
<td>1000 km</td>
<td>Continent</td>
<td></td>
</tr>
<tr>
<td>10,000 km</td>
<td>Planet</td>
<td></td>
</tr>
</tbody>
</table>
The Internet

- global system of interconnected WANs and LANs
- open, system-independent, no global control

[Tanenbaum, Computer Networks]
Interconnection of Subnetworks

[Tanenbaum, Computer Networks]
History of the Internet

- 1961: Packet Switching Theory
  - Leonard Kleinrock, MIT, “Information Flow in Communication Nets”
- 1962: Concept of a “Galactic Network”
- 1965: Predecessor of the Internet
  - Analog modem connection between 2 computers in the USA
- 1967: Concept of the “ARPANET”
  - Concept of Larry Roberts
- 1969: 1st node of the “ARPANET”
  - at UCLA (Los Angeles)
  - end 1969: 4 computers connected
ARPANET (a) December 1969   (b) July 1970
(c) March 1971    (d) April 1972 (e) September 1972
Internet ~2005
An Open Network Architecture

- Concept of Robert Kahn (DARPA 1972)
  - Local networks are autonomous
    - independent
    - no WAN configuration
  - packet-based communication
  - "best effort" communication
    - if a packet cannot reach the destination, it will be deleted
    - the application will re-transmit
  - black-box approach to connections
    - black boxes: gateways and routers
    - packet information is not stored
    - no flow control
  - no global control

- Basic principles of the Internet
# Protocols of the Internet

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Telnet, FTP, HTTP, SMTP (E-Mail), ...</td>
</tr>
<tr>
<td>Transport</td>
<td>TCP (Transmission Control Protocol)  UDP (User Datagram Protocol)</td>
</tr>
<tr>
<td>Host-to-Network</td>
<td>LAN (e.g. Ethernet, W-Lan etc.)</td>
</tr>
</tbody>
</table>
TCP/IP Layers

1. Host-to-Network
   - Not specified, depends on the local network, e.g. Ethernet, WLAN 802.11, PPP, DSL

2. Routing Layer/Network Layer (IP - Internet Protocol)
   - Defined packet format and protocol
   - Routing
   - Forwarding

3. Transport Layer
   - TCP (Transmission Control Protocol)
     • Reliable, connection-oriented transmission
     • Fragmentation, Flow Control, Multiplexing
   - UDP (User Datagram Protocol)
     • Hands packets over to IP
     • Unreliable, no flow control

4. Application Layer
   - Services such as TELNET, FTP, SMTP, HTTP, NNTP (for DNS), …
   - Peer-to-peer networks
Reference Models: OSI versus TCP/IP

**OSI**

- 7 Application
- 6 Presentation
- 5 Session
- 4 Transport
- 3 Network
- 2 Data link
- 1 Physical

**TCP/IP**

- Application
- Transport
- Internet
- Host-to-network

*Not present in the model*

(Aus Tanenbaum)
Network Interconnections

<table>
<thead>
<tr>
<th>Layer</th>
<th>Interconnection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Application gateway</td>
</tr>
<tr>
<td>Transport</td>
<td>Transport gateway</td>
</tr>
<tr>
<td>Network</td>
<td>Router</td>
</tr>
<tr>
<td>Data link</td>
<td>Bridge, switch</td>
</tr>
<tr>
<td>Physical</td>
<td>Repeater, hub</td>
</tr>
</tbody>
</table>

[Tanenbaum, Computer Networks]
Example: Routing between LANs

Stevens, TCP/IP Illustrated
Data/Packet Encapsulation

Stevens, TCP/IP Illustrated
IPv4-Header (RFC 791)

- Version: 4 = IPv4
- IHL: IP header length
  - in 32 bit words
  - (>5)
- Type of service
  - optimize delay,
    throughput, reliability,
    monetary cost
- Checksum (only IP-header)
- Source and destination IP-address
- Protocol identifies protocol
  - e.g. TCP, UDP, ICMP, IGMP
- Time to Live:
  - maximal number of hops
IP addresses and Domain Name System

- **IP addresses**
  - every interface in a network has a unique world wide IP address
  - separated in Net-ID and Host-ID
  - Net-ID assigned by Internet Network Information Center
  - Host-ID by local network administration

- **Domain Name System (DNS)**
  - replaces IP addresses like 132.230.167.230 by names, e.g. falcon.informatik.uni-freiburg.de and vice versa
  - Robust distributed database
Internet IP Addresses
Classfull Addresses until 1993

- Classes A, B, and C
- D for multicast; E: “reserved”
Classless IPv4-Addresses

- **Until 1993 (deprecated)**
  - 5 classes marked by Präfix
  - Then sub-net-id prefix of fixed length and host-id

- **Since 1993**
  - Classless Inter-Domain-Routing (CIDR)
  - Net-ID and Host-ID are distributed flexibly
  - E.g.
    - Network mask /24 or 11111111.11111111.11111111.00000000
    - denotes, that IP-address
      - 10000100. 11100110. 10010110. 11110011
      - consists of network 10000100. 11100110. 10010110
      - and host 11110011

- **Route aggregation**
  - Routing protocols BGP, RIP v2 and OSPF can address multiple networks using one ID
    - Z.B. all Networks with ID 10010101010* can be reached over host X
Routing Tables and Packet Forwarding

- **IP Routing Table**
  - contains for each destination the address of the next gateway
  - destination: host computer or sub-network
  - default gateway

- **Packet Forwarding**
  - IP packet (datagram) contains start IP address and destination IP address
    - if destination = my address then hand over to higher layer
    - if destination in routing table then forward packet to corresponding gateway
    - if destination IP subnet in routing table then forward packet to corresponding gateway
    - otherwise, use the default gateway
IP Packet Forwarding

- IP Packet (datagram) contains...
  - TTL (Time-to-Live): Hop count limit
  - Start IP Address
  - Destination IP Address

Packet Handling
- Reduce TTL (Time to Live) by 1
- If TTL ≠ 0 then forward packet according to routing table
- If TTL = 0 or forwarding error (buffer full etc.):
  - delete packet
  - if packet is not an ICMP Packet then
    - send ICMP Packet with
      - start = current IP Address
      - destination = original start IP Address
Introduction to Future IP

- IP version 6 (IP v6 – around July 1994)
- Why switch?
  - rapid, exponential growth of networked computers
  - shortage (limit) of the addresses
  - new requirements towards the Internet infrastructure (streaming, real-time services like VoIP, video on demand)
- evolutionary step from IPv4
- interoperable with IPv4
Capabilities of IP

- dramatic changes of IP
  - Basic principles still appropriate today
  - Many new types of hardware
  - Scale of Internet and interconnected computers in private LAN

- Scaling
  - Size - from a few tens to a few tens of millions of computers
  - Speed - from 9,6Kbps (GSM) to 10Gbps (Ethernet)
  - Increased frame size (MTU) in hardware
IPv6-Header (RFC 2460)

- Version: 6 = IPv6
- Traffic Class
  - for QoS (priority)
- Flow Label
  - QoS or real-time
- Payload Length
  - size of the rest of the IP packet
- Next Header (IPv4: protocol)
  - e.g. ICMP, IGMP, TCP, EGP, UDP, Multiplexing, ...
- Hop Limit (Time to Live)
  - maximum number of hops
- Source Address
- Destination Address
  - 128 bit IPv6 address
Static and Dynamic Routing

- Static Routing
  - Routing table created manually
  - used in small LANs

- Dynamic Routing
  - Routing table created by Routing Algorithm
  - Centralized, e.g. Link State
    - Router knows the complete network topology
  - Decentralized, e.g. Distance Vector
    - Router knows gateways in its local neighborhood
Intra-AS Routing

- Routing Information Protocol (RIP)
  - Distance Vector Algorithmus
  - Metric = hop count
  - exchange of distance vectors (by UDP)
- Interior Gateway Routing Protocol (IGRP)
  - successor of RIP
  - different routing metrics (delay, bandwidth)
- Open Shortest Path First (OSPF)
  - Link State Routing (every router knows the topology)
  - Route calculation by Dijkstra’s shortest path algorithm
Distance Vector Routing Protocol

- Distance Table data structure
  - Each node has a
    - Line for each possible destination
    - Column for any direct neighbors
- Distributed algorithm
  - Each node communicates only with its neighbors
- Asynchronous operation
  - Nodes do not need to exchange information in each round
- Self-terminating
  - Exchange unless no update is available

Distance Table for A

<table>
<thead>
<tr>
<th>from A</th>
<th>via</th>
<th>Routing Table entry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
<td>B</td>
</tr>
<tr>
<td>E</td>
<td>8</td>
<td>E</td>
</tr>
</tbody>
</table>

Distance Table for C

<table>
<thead>
<tr>
<th>from C</th>
<th>via</th>
<th>Routing Table entry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>6</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>7</td>
<td>D</td>
</tr>
</tbody>
</table>

Distance Table Data Structure

- Each node has a
  - Line for each possible destination
  - Column for any direct neighbors

Distributed Algorithm

- Each node communicates only with its neighbors

Asynchronous Operation

- Nodes do not need to exchange information in each round

Self-Terminating

- Exchange unless no update is available
Distance Vector Routing Example

![Diagram of a network graph with nodes A, B, C, D, and E connected by edges with weights 1, 3, 5, 7, and 9.]

<table>
<thead>
<tr>
<th>from A to</th>
<th>via</th>
<th>entry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>E</td>
<td>7</td>
<td>4</td>
</tr>
</tbody>
</table>
## Distance Vector Routing

### Network Graph

```
A --- 1 --- B
   | 5   |
   v 1  
C --- 7 --- D
   | 1   |
   v 3  
E
```

### Distance Matrix

<table>
<thead>
<tr>
<th>from A to</th>
<th>via</th>
<th>entry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>from B to</th>
<th>via</th>
<th>entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>-</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>from C to</th>
<th>via</th>
<th>entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>-</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>
Distance Vector Routing

<table>
<thead>
<tr>
<th>from B to</th>
<th>via</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>from C to</th>
<th>via</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>from B to</th>
<th>via</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td>13</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>from C to</th>
<th>via</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>B</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>E</td>
<td>-</td>
<td>13</td>
</tr>
</tbody>
</table>

1. **From B to**:
- A: 1
- C: 5
- D: 13
- E: 6

2. **From C to**:
- A: 3
- B: 5
- D: 6
- E: 13

3. **From E to**:
- B: 7
- C: 1
- D: 13
- E: 1
“Count to Infinity” - Problem

- Good news travels fast
  - A new connection is quickly at hand
- Bad news travels slowly
  - Connection fails
  - Neighbors increase their distance mutually
  - "Count to Infinity" Problem
### “Count to Infinity” - Problem

<table>
<thead>
<tr>
<th></th>
<th>from A</th>
<th>via</th>
<th>Routing Table entry</th>
<th>from B</th>
<th>via</th>
<th>Routing Table entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>to</td>
<td>B</td>
<td>2</td>
<td>B</td>
<td>A</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>3</td>
<td>B</td>
<td>C</td>
<td>5</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>from A</th>
<th>via</th>
<th>Routing Table entry</th>
<th>from B</th>
<th>via</th>
<th>Routing Table entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>to</td>
<td>B</td>
<td>2</td>
<td>B</td>
<td>A</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>7</td>
<td>B</td>
<td>C</td>
<td>5</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>from A</th>
<th>via</th>
<th>Routing Table entry</th>
<th>from B</th>
<th>via</th>
<th>Routing Table entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>to</td>
<td>B</td>
<td>2</td>
<td>B</td>
<td>A</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>7</td>
<td>B</td>
<td>C</td>
<td>9</td>
<td>-</td>
</tr>
</tbody>
</table>
Link-State Protocol

- Link state routers
  - exchange information using Link State Packets (LSP)
  - each node uses shortest path algorithm to compute the routing table

- LSP contains
  - ID of the node generating the packet
  - Cost of this node to any direct neighbors
  - Sequence-no. (SEQNO)
  - TTL field for that field (time to live)

- Reliable flooding (Reliable Flooding)
  - current LSP of each node are stored
  - Forward of LSP to all neighbors
    - except to be node where it has been received from
  - Periodically creation of new LSPs
    - with increasing SEQNO
  - Decrement TTL when LSPs are forwarded
Inter-AS: BGPv4 (Border Gateway Protocol)

- **de facto standard**

- **Path-Vector-Protocol**
  - like Distance Vector Protocol
    - store whole path to the target
  - each Border Gateway advertizes to all its neighbors (peers) the complete path to the target (per TCP)

- If gateway X sends the path to the peer-gateway W
  - then W can choose the path or not
  - optimization criteria
    - cost, policy, etc.
  - if W chooses the path of X, it publishes
    - Path(W,Z) = (W, Path(X,Z))

- **Remark**
  - X can control incoming traffic using advertisements
  - all details hidden here
BGP-Routing Table Size
1994-2013

http://bgp.potaroo.net/as1221/bgp-active.html
Network Congestion

- (Sub-)Networks have limited bandwidth
- Injecting too many packets leads to
  - network congestion
  - network collapse
Congestion and capacity

Packets delivered vs Packets sent diagram:
- Maximum carrying capacity of subnet
- Perfect
- Desirable
- Congested
# Congestion Prevention

<table>
<thead>
<tr>
<th>Layer</th>
<th>Policies</th>
</tr>
</thead>
</table>
| Transport| • Retransmission policy  
          | • Out-of-order caching policy  
          | • Acknowledgement policy  
          | • Flow control policy  
          | • Timeout determination |
| Network  | • Virtual circuits versus datagram inside the subnet  
          | • Packet queueing and service policy  
          | • Packet discard policy  
          | • Routing algorithm  
          | • Packet lifetime management |
| Data link| • Retransmission policy  
          | • Out-of-order caching policy  
          | • Acknowledgement policy  
          | • Flow control policy |
IP Routers drop packets
- Tail dropping
- Random Early Detection
Random early detection (RED)

- Packet dropping probability grows with queue length.
- Fairer than just “tail dropping”: the more a host transmits, the more likely it is that its packets are dropped.

![Diagram showing MaxThreshold, MinThreshold, AvgLen, and P(drop) vs AvgLen with MaxP at 1.0]
The Transport Layer

- **TCP** (Transmission Control Protocol)
  - connection-oriented
  - delivers a stream of bytes
  - reliable and ordered
- **UDP** (User Datagram Protocol)
  - delivery of datagrams
  - connectionless, unreliable, unordered
TCP vs. UDP

- TCP reduces data rate
- UDP does not!
UDP-Header

- Port addresses
  - for parallel UDP connections
- Length
  - data + header length
- Checksum
  - for header and data

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>31</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>Source</td>
<td>Destination</td>
</tr>
<tr>
<td>Port</td>
<td>Port</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>Length</td>
<td>Checksum</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td>data octets ...</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
The Transmission Control Protocol (TCP)

- Connection-oriented
- Reliable delivery of a byte stream
  - fragmentation and reassembly (TCP segments)
  - acknowledgements and retransmission
- In-order delivery, duplicate detection
  - sequence numbers
- Flow control and congestion control
  - window-based (receiver window, congestion window)

- challenge: IP (network layer) packets can be dropped, delayed, delivered out-of-order ...
**TCP-Header**

- **Sequence number**
  - number of the first byte in the segment
  - bytes are numbered modulo $2^{32}$

- **Acknowledge number**
  - activated by ACK-Flag
  - number of the next data byte
    - $= \text{last sequence number} + \text{last amount of data}$

- **Port addresses**
  - for parallel TCP connections

- **TCP Header length**
  - data offset

- **Check sum**
  - for header and data
TCP Connections

- Connection establishment and teardown by 3-way handshake

**Connection establishment**

- Host 1
  - `syn seq=x`
  - `syn ack=x+1 seq=y`
  - `ack=y+1 seq=x+1 [data]`

- Host 2

**Connection termination**

- Host 1
  - `fin, seq=x`
  - `ack=x+1`

- Host 2
  - `fin, seq=y`
  - `ack=y+1`
Flow control and congestion control

[Tanenbaum, Computer Networks]

(a)

(b)
Flow Control

acknowledgements and window management

- Application does a 2K write
  - Sender
  - Receiver
  - ACK = 2048 WIN = 2048
  - ACK = 4096 WIN = 0
  - Application reads 2K
- Sender is blocked
- Sender may send up to 2K
- Receiver's buffer
  - 0 4K
  - Empty
  - Full
  - 1K 2K
Retransmissions

- Retransmissions are triggered, if acknowledgements do not arrive... but how to decide that?

- Measurement of the round trip time (RTT)
Retransmissions and RTT

Sender

Round Trip Time

Receiver

Retransmission after timeout
Estimation of the Round Trip Time (RTT)

- If no acknowledgement arrives before expiry of the **Retransmission Timeout (RTO)**, the packet will be retransmitted
  - RTT not predictable, fluctuating

**RTO derived from RTT estimation:**
- RFC 793: \( M := \text{last RTT measurement} \)
  - \( \text{RTT} \leftarrow \alpha \text{RTT} + (1-\alpha) M, \) where \( \alpha = 0.9 \)
  - \( \text{RTO} \leftarrow \beta \text{RTT}, \) where \( \beta = 2 \)

- Alternative by Jacobson 88 (using the deviation \( D \)):
  - \( D \leftarrow \alpha' D + (1-\alpha') |\text{RTT} - M| \)
  - \( \text{RTT} \leftarrow \alpha \text{RTT} + (1-\alpha) M \)
  - \( \text{RTO} \leftarrow \text{RTT} + 4D \)
TCP - Algorithm of Nagle

- How to ensure
  - small packages are shipped fast
  - yet, large packets are preferred

- Algorithm of Nagle
  - Small packets are not sent, as long as acks are still pending
    - Package is small, if data length <MSS
  - when the acknowledgment of the last packet arrives, the next one is sent

- Example:
  - terminal versus file transfer versus ftp

- Feature: self-clocking:
  - Quick link = many small packets
  - slow link = few large packets
Congestion revisited

- IP Routers drop packets
- TCP has to react, e.g. lower the packet injection rate
Congestion revisited

From a transport layer perspective:

no ACKs received
Data rate adaption and the congestion window

- Sender does not use the maximum segment size in the beginning

- Congestion window (cwnd)
  - used on the sender size
  - sending window: min \{wnd, cwnd\} (wnd = receiver window)
  - S: segment size
  - Initialization:
    - cwnd ← S
  - For each received acknowledgement:
    - cwnd ← cwnd + S
  - ...until a packet remains unacknowledged
Slow Start of TCP Tahoe

- Slow start
- Timeout
- Threshold
TCP Tahoe’s slow start

- TCP Tahoe, Jacobson 88:
  - Congestion window (cwnd)
  - Slow Start Threshold (ssthresh)
  - S = maximum segment size

- Initialization (after connection establishment):
  - cwnd ← S  ssthresh ← 65535

- If a packet is lost (no acknowledgement within RTO):
  - multiplicative decrease of ssthresh
    cwnd ← S  ssthresh ← \( \max\{2S, \frac{\min\{cwnd, \text{wnd}\}}{2}\} \)

- If a segment is acknowledged and \( cwnd \leq ssthresh \) then
  - slow start:  cwnd ← cwnd + S

- If a segment is acknowledged and \( cwnd > ssthresh \), then

  \[
  cwnd \leftarrow cwnd + \frac{S}{cwnd}
  \]
Fast Retransmit and Fast Recovery

- TCP Tahoe [Jacobson 1988]:
  - If only one packet is lost
    • retransmit and use the rest of the window
  - Slow Start
  - Fast Retransmit
    • after three duplicate ACKs, retransmit Packet, start with Slow Start

- TCP Reno [Stevens 1994]
  - After Fast Retransmit:
    • ssthresh ← min(wnd,cwnd)/2
    • cwnd ← ssthresh + 3 S
  - Fast recovery after Fast retransmit
    • Increase window size by each single acknowledgement
    • cwnd ← cwnd + S
  - Congestion avoidance: if P+x is acknowledged:
    • cwnd ← ssthresh
The AIMD principle

- TCP uses basically the following mechanism to adapt the data rate $x$ (#packets sent per RTT):

  - Initialization: $x \leftarrow 1$

  - on packet loss: multiplicative decrease (MD) $x \leftarrow x/2$

  - if the acknowledgement for a segment arrives, perform additive increase (AI) $x \leftarrow x + 1$
AIMD

additive increase

multiplicative decrease
Throughput and Latency

- **Congested situation (cliff):**
  - high load
  - low throughput
  - all data packets are lost

- **Desired situation (knee):**
  - high load
  - high throughput
  - few data packets get lost
Vector diagram for 2 participants

- Data rate of A
- Data rate of B
- b: max. available bandwidth
- Fairness
- Efficiency
- Optimal data rate
AIAD Additive Increase/ Additive Decrease
MIMD: Multiplicative Incr./ Multiplicative Decrease
AIMD: Additively Increase/Multiplicatively Decrease
TCP - Conclusion

- Connection-oriented, reliable, in-order delivery of a byte stream
- Flow control and congestion control
  - Fairness among TCP streams
  - Unfair behavior of other protocols, e.g. UDP
  - Impact on latency
  - Tweaking the congestion avoidance mechanism has an impact on other applications
Peer-to-Peer Networks
13 Internet – The Underlay Network

Christian Ortolf
Technical Faculty
Computer-Networks and Telematics
University of Freiburg